Discrete Applied Mathematics 156 (2008) 2480–2492

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Identical part production in cyclic robotic cells: Concepts, overview and open questions

Nadia Brauner

G-SCOP, UJF, 46 avenue Felix Viallet, 38031 Grenoble Cedex, France

ARTICLE INFO

Article history:
Received 18 September 2006
Received in revised form 27 June 2007
Accepted 6 March 2008
Available online 25 April 2008

Keywords:
Scheduling
Flow-shop
Material handling system
Cyclic production

ABSTRACT

Robotic cells consist of a flow-shop with a robot for material handling. A single part is to be produced cyclically and the objective is to minimize production rate. This document introduces basic concepts and tools for dealing with cyclic production. In particular, it concentrates on k-cycles which are production cycles where exactly k parts enter and leave the cell. One defines the cycle function $\mathcal{K}$ which is the smallest value of k so that the set of all k-cycles up to size $\mathcal{K}$ contains an optimal cycle for all instances. Known results and conjectures on these functions are given for the classical case where parts can remain on the machine waiting for the robot and for the no-wait case where parts have to be removed from the machine as soon as their processing is finished.

Robotic flow-shops consist of m machines served by a single central robot. They were first introduced by [4] and studied by [40]. In [4], a line for machining castings for truck differential assemblies is described in the form of a 3-machine robotic cell where a robot has to transfer heavy mechanical parts between large machines. The system contains a conveyor belt for incoming parts and another one for outgoing parts. In this particular system the robot is not able to traverse the conveyor. Therefore, the movement of the robot from the output to the input station has to traverse the entire cell.

The original application has the form of a flow-shop. However, robotic cells may have very flexible configurations. The robot can easily access the machines thus producing a large variety of products in form of a job-shop. But it is known that the robotic scheduling problem is already NP-hard for a flow-shop with $m \geq 3$ machines and two or more different part types [28]. The case of the m-machine robotic cell in which one wants to produce a single batch of identical parts remains of interest. We shall mainly concentrate on this case. The robot may have unit capacity, as will be the case in our model, or one may have two-unit robots [42,41] or a multi-robot cell [32,30]. Robotic cells with buffers at the machine have been studied in [24,13]. Operation and/or process flexibility (the order of the operations or the assignment of the operations to the machines are not fixed) was recently studied in [3,27,26]. We concentrate here on the no-buffer case where parts are either on a machine or transported by the robot and no flexibility on operations or on the process is allowed. A survey on general robotic cells can be found in [18,23].

This document gives a state of the art on cyclic scheduling of identical parts in robotic cells. It describes classical configurations of robotic cells (Section 1) and introduces basic concepts and tools for dealing with cyclic production (Section 2). Then it describes known results for the classical case where parts can remain on the machine waiting for the robot (Section 3) and for the no-wait case where the parts have to be removed from the machine as soon as their processing is finished (Section 4). Then a detailed list of open questions is given (Section 5).

1. Robotic cells

The m machines of a robotic cell are denoted by $M_1, M_2 \ldots M_m$ and we add two auxiliary machines, M_0 for the input station IN and M_{m+1} for the output station OUT (Fig. 1). Raw material for the parts to be produced is available in unlimited quantity

E-mail address: nadia.brauner@g-scop.inpg.fr.

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.03.021

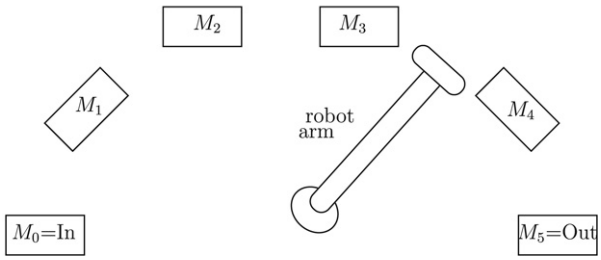

Fig. 1. Robotic cell with $m = 4$ machines.

Table 1
Two different expressions of the production constraints

	Processing	Waiting policy	Another description
No-wait	p_i	0	$\underline{p_{ij}} = \overline{p_{ij}}$
HSP	in $[\underline{p_{ij}}, \overline{p_{ij}}]$	0	$\underline{p_{ij}} \leq \overline{p_{ij}}$
Unbounded	p_i	Unbounded	$\overline{p_{ij}} = +\infty$

at M_0. The central robot can handle a single unit at a time. A part is picked up at M_0 and transferred in succession to M_1, $M_2 \ldots M_m$, where it is machined in this order until it finally reaches the output station M_{m+1}. At M_{m+1}, the finished parts can be stored in unlimited amounts. We focus on the classical case as in [40], where machines $M_1, M_2 \ldots M_m$ are without buffer facility. In this case, the robot, with unit capacity, has to be empty whenever it wants to pick up a part at M_h ($h = 0, 1 \ldots m$).

Consider an instance I of an m-machine robotic cell. Different cell configurations have been studied, depending mainly on production constraints (Section 1.1) and on the metric for travel times (Section 1.2).

1.1. The production constraints

Processing starts as soon as a part is loaded on a machine. The *processing time* represents the minimum time a part must remain on a machine. If all parts are different, p_{ij} denotes the processing time of part j on machine M_i. If one wants to produce one large batch of *identical parts*, the processing times of the parts on machine M_i are $p_i = p_{ij}$. In the *balanced* case, all processing times are equal, *i.e.*, $p_i = p$ for all i.

Once the part is finished, two policies may apply. In the *no-wait* case, the part must be removed immediately from the machine and transferred to the following machine. In the *unbounded* case, the part can remain on the machine waiting for the robot.

A classical extension of those two cases is the so-called Hoist Scheduling Problem (HSP) for which the processing policy is different. For the preceding two cases, the processing time is fixed. For the HSP, the processing time is described by an interval, and the no-wait policy applies. This means that the time part j may remain on machine M_i lays in the interval $[\underline{p_{ij}}, \overline{p_{ij}}]$ (see Table 1). This applies to chemical treatments were the machines correspond to chemical baths. The HSP is NP-hard even for identical parts and very simple (additive) configurations of cells [20]. There exists a wide literature on this problem that we will not develop here (see e.g. [5]).

Denote by ϵ the time to load a part onto a machine from the robot or to unload a part from a machine onto the robot.

1.2. The travel metric of the robot

We shall consider different classical metrics for travel times of the robot depending on the physical configuration of the cell and on the characteristics of the robot. Denote by $\delta_{h,h'}$, the travel time of the robot (empty or loaded) from M_h to $M_{h'}$. The following natural, and in practice desirable, assumptions are made [14]:

- the travel time from a machine to itself is zero, that is, $\delta_{h,h} = 0$;
- the travel times satisfy the triangle inequality, that is, $\delta_{h,k} + \delta_{k,h'} \geq \delta_{h,h'}$ for all h, k and h';
- The travel times are symmetric, that is $\delta_{h,h'} = \delta_{h',h}$ for all h and h'.

Travel times verifying those three assumptions are called *general* (or sometimes "euclidean", e.g. in [23]). Special configurations of cells have been studied. Table 2 summarizes the most classical ones which we now define in detail.

For *additive* times, to travel between distant machines, the robot passes through all intermediate machines and its speed is constant. This metric is the most popular since, in practice, it is applicable if the machines are on a circle or on a line and if the cell is dense (the robot does not have time to speed up between distant machines). In this case, one has the triangle equality $\delta_{h,h'} = \delta_{h,k} + \delta_{k,h'} = \sum_{k=h}^{h'-1} \delta_{k,k+1}$ for any $h < h'$. By symmetry, this also defines $\delta_{h,h'}$ for $h > h'$. Some authors

Table 2
Some classical metrics for the robot travel times

	$\delta_{h,h+1}$ not constrained	regular ($\delta_{h,h+1} = \delta$)
General	$\delta_{h,h} = 0$; $\delta_{h,h'} = \delta_{h',h}$ $\delta_{h,h'} \leq \delta_{h,k} + \delta_{k,h'}$	$\delta_{h,h+1} = \delta$
Additive	$\delta_{h,h'} = \delta_{h,k} + \delta_{k,h'}$ for $h < k < h'$	$\delta_{h,h'} = \lvert h' - h \rvert \delta$
Circular	Shortest path along the circle	$\delta_{h,h'} = \min(\lvert h - h' \rvert, m + 1 - \lvert h - h' \rvert)$
Constant	$\delta_{h,h'} = \delta$	

(e.g. [29,35]) consider an extension of this case having a constant gain γ when travelling between distant machines, *i.e.*, $\delta_{h,h'} = \sum_{k=h}^{h'-1} \delta_{k,k+1} - (h' - h - 1)\gamma$ for $h < h'$. We assume $\gamma = 0$.

In *circular* cells, the input and the output stations coincide, *i.e.* $M_0 = M_{m+1}$. Thus the robot chooses the shortest path along the circle formed by the machines. Then, the travel times verify $\delta_{h,h'} = \min(\sum_{k=h}^{h'-1} \delta_{k,k+1}, \sum_{k=h'}^{m} \delta_{k,k+1} + \sum_{k=0}^{h-1} \delta_{k,k+1})$ for any $h < h'$.

In the *regular* case, machines are equidistant and we denote $\delta_{h,h+1} = \delta$. This constraint can be added to additive (as in the seminal paper [40]) or to circular cells.

For *constant* travel times (introduced in [22]), δ is the time for the robot to travel between any two distinct machines M_h and $M_{h'}$: $\delta_{h,h'} = \delta$. The interest of this metric is that it is simpler to study than the others but it seems to have the same properties as the general additive metric.

2. Activities and k-cycles

The robotic scheduling problem is already NP-hard for a flow-shop with $m \geq 3$ machines and two or more different part types [28]. Therefore, we concentrate on the interesting case of the m-machine robotic cell in which one wants to produce identical parts. Then the problem reduces to finding the optimal strategy for the robot moves in order to obtain the maximal throughput rate for this unique part.

In [23], the authors prove that there always exists a cyclic production that is optimal. Therefore, we consider cyclic robot moves for the production process of parts and define a *k-cycle* as a production cycle of exactly k parts. It can be described as a sequence of robot moves where exactly k parts enter the system at M_0, k parts leave the system at M_{m+1} and each time the robot executes the k-cycle, the system returns to the same state, *i.e.* the same machines are loaded, the same machines are empty and the robot returns to the starting position. To describe k-cycles we use the concept of *activities* [21]. The activity A_h ($h = 0, 1 \ldots m$) consists of the following sequence:

- the idle robot takes a part from M_h;
- the robot travels with this part from M_h to M_{h+1};
- the robot loads this part onto M_{h+1}.

Note, that many sequences are not feasible, e.g. $(\ldots A_0, A_0 \ldots)$, since the robot carries a part to M_1 which is occupied. In [21], the authors characterize k-cycles as follows: A k-cycle C_k is a sequence of activities, in which each activity occurs exactly k times and between two consecutive (in a cyclic sense) occurrences of $A_h (h = 1, 2 \ldots m - 1)$ there is exactly one occurrence of A_{h-1} and exactly one occurrence of A_{h+1}.

We represent a k-cycle C_k as in Fig. 2.The horizontal axis represents time. The vertical axis represents the cell. The graph indicates the position of the robot in the cell while executing the cycle. Dashed lines are empty robot moves and plain lines are loaded robot moves, the loading and unloading processes or the waiting times of the robot at the machines. Let us illustrate this with an example. In a 3-machine regular additive cell, consider the 1-cycle $C = (A_0 A_2 A_1 A_3)$. Let I be the following instance:

$$\delta = 1; \qquad \epsilon = 0; \qquad p_1 = 6; \qquad p_2 = 9; \qquad p_3 = 6.$$

At the beginning of the cycle C, machine M_2 is loaded and machines M_1 and M_3 are empty. We suppose that, at time 0, the part has been on M_2 for 6 time units. At time 9, on Fig. 2, the robot is at machine M_3 and is waiting one time unit for the part to be ready in order to execute activity A_3. One can observe that the cycle does not repeat identically. In this example, if the part was on M_2 for 5.5 time units (instead of 6), the cycle would have repeated identically. However, the mean cycle time (14.5 in both case) does not seem to depend on the initial state (ergodicity). Let $T(C_k)$ be the long run average execution time of the k-cycle C_k.

We call $T(C_k)$ the *cycle time* and $T(C_k)/k$ the *cycle length*. The *throughput rate* is defined by $k/T(C_k)$. Thus the ρ-cycle C_ρ is *optimal* if it maximizes throughput rate or equivalently minimizes cycle length $T(C_k)/k$ over the set of all possible k-cycles ($k = 1, 2, 3 \ldots$). A set of cycles S is said to be *dominant* if, for any instance, there exists a cycle of S that is optimal.

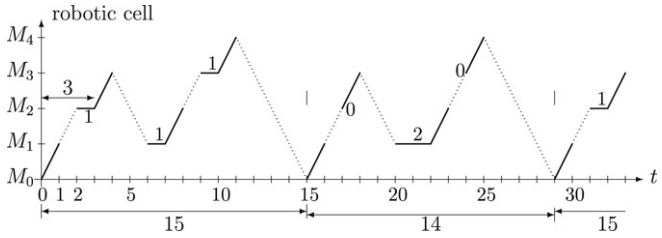

Fig. 2. Representation of the 1-cycle $C = (A_0A_2A_1A_3)$ for the instance I.

2.1. Dominant sets of cycles

Ideally, one would like to determine, for a given instance, an optimal k-cycle. However, this is so far not possible, except for very particular cases, for instance for very slow or for very fast robots compared to processing times. In [40] the authors proposed the following conjecture

1-*cycle Conjecture* [40]: The set of 1-cycles is dominant. This conjecture is valid for 2-machine cells (see Section 2.4) and unbounded 3-machine cells [21,9]. However it is false for no-wait 3-machine cells [1] and unbounded regular additive or constant 4-machine cells [11,23]. It has been replaced by the following conjecture:

Agnetis' Conjecture [1]: The set of k-cycles with $k \leq m-1$ is dominant.

Note that this conjecture was originally formulated for additive no-wait cells. Let S_K be the set of all k-cycles with $1 \leq k \leq K$. We are interested in the minimal dominant set $S_{\mathcal{K}}$, *i.e.*, $S_{\mathcal{K}}$ is dominant and no $S_{\mathcal{K}'}$ is dominant with $\mathcal{K}' < \mathcal{K}$. We can expect that $\mathcal{K} = \mathcal{K}(m)$ is a function of the number of machines m. We call $\mathcal{K}(m)$ the *cycle function*.

Finding the optimal production cycle can be decomposed into three sub-problems that we address in the following sections:

(P1) Determine $\mathcal{K}$, *i.e.*, find its constant value if it is indeed constant, lower bounds, finite upper bounds…
(P2) Determine the complexity of finding the best cycle in S_k (where k can be any number between 1 and $\mathcal{K}$).
(P3) Determine the performance of S_k (where k can be any number between 1 and $\mathcal{K}$) defined by

$$\mathcal{P}(k) = \max_{\text{instances}} \frac{\text{best cycle length in } S_k}{\text{best cycle length in } S_{\mathcal{K}}}.$$

Those three problems have been widely studied especially (P2) and (P3) for $k = 1$ when the 1-cycle Conjecture was still open. In the following sections, we review the state of the art for those problems for the unbounded case (Section 3) and for the no-wait case (Section 4).

2.2. Bounds

In this section, we introduce several bounds on the cycle time. Consider a k-cycle C_k. We first introduce some inequalities that only depend on the cycle and not on the configuration of the cell. Let $m_h(C_k)$ be the number of times the robot travels between machines M_h and M_{h+1} in both directions during one execution of the k-cycle C_k and let $|S|_{C_k}$ be the number of occurrences of the sequence of activities S in C_k and let $u_i(C_k) = |A_{i-1}A_i|_{C_k}$. For simplicity, we drop C_k in those notations when no confusion is possible. If the robot never makes any dummy moves, one has [11,9]:

$$m_0 = 2k \tag{1}$$
$$m_m = 2k \tag{2}$$
$$m_1 = 4k - 2u_1 \tag{3}$$
$$m_{m-1} = 4k - 2u_m \tag{4}$$
$$m_2 \geq 4k - 2u_2 - 2|A_1A_0A_2| \tag{5}$$
$$m_{m-2} \geq 4k - 2u_{m-1} - 2|A_{m-2}A_mA_{m-1}|. \tag{6}$$

The following inequalities are often used as an optimality criterion. The intuition is that the lower bound is the time elapsed between two successive loadings of machine M_h (see Fig. 3). Since this happens k times in a cycle, we have a multiplicative factor k.

$$T(C_k) \geq k(\delta_{h,h+1} + \delta_{h+1,h-1} + \delta_{h-1,h} + p_h + 4\epsilon) \quad \forall h = 1, 2 \ldots m. \tag{7}$$

In the additive case it becomes,

$$T(C_k) \geq k(2\delta_{h,h+1} + 2\delta_{h-1,h} + p_h + 4\epsilon) \quad \forall h = 1, 2 \ldots m \tag{8}$$

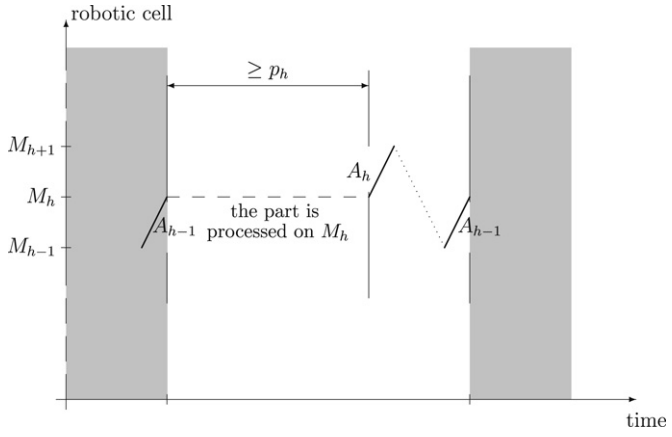

Fig. 3. Illustration of inequalities (8) and (9).

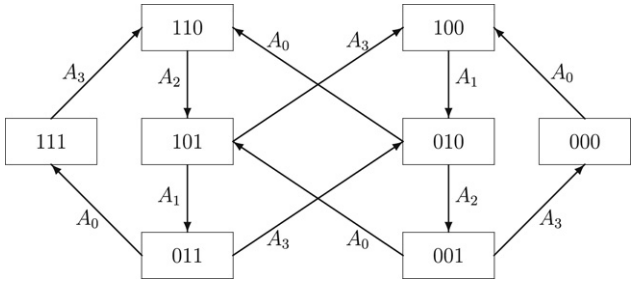

Fig. 4. The state graph G_3.

and in the constant case one has,

$$T(C_k) \geq k(3\delta + p_h + 4\epsilon) \quad \forall h = 1, 2 \ldots m \tag{9}$$

One of the interests of the constant case is the fact that travel time can be expressed easily: one has the following equality for travel time $T_T(C_k)$ of a k-cycle C_k:

$$T_T(C_k) = 2k(m+1)\delta - \sum_{i=1}^{m} u_i\delta. \tag{10}$$

The intuition for this equality is that between two activities, one has a time δ if and only if the two activities are not consecutive, *i.e.* they do not participate in a u_i. This equality is proved for $k = 1$ in [22].

2.3. State graphs

At each instant in a production cycle, it is possible to calculate the part/machine incidence vector: machine M_h is loaded if and only if the next occurrence of A_h arrives before the next occurrence of A_{h-1}. For instance, at the beginning of the execution of the cycle $(A_0, A_4, A_6, A_7, A_5, A_3, A_2, A_1)$, the part/machine incidence vector is (0, 1, 1, 1, 0, 1, 0).

In the *state graph* G_m associated with an m-machine robotic cell, each vertex is a part/machine incidence vector that represents the state of the cell. Therefore, G_m has 2^m vertices. Arcs represent the activities of the robot to pass from one state to another state. The state graph G_3 is given in Fig. 4. In G_3, '100' represents the state of the system with machine M_1 loaded and machines M_2 and M_3 empty. To go from state '100' to state '010', the robot transfers a part from M_1 to M_2, executing activity A_1. Each k-cycle corresponds uniquely to a cycle of length $k(m+1)$ in the graph. For instance, the 1-cycle $(A_0A_3A_1A_2)$ corresponds to the sequence of vertices '001', '101', '100', '010'.

State graphs allow us to find the number of k-cycles in an m-machine cell [15] (note that e.g. a 2-cycle might be the repetition of twice the same 1-cycle). The algorithm is based on the traversal of the state graph. A label is attached to each cycle. This label is the inverse of the number of times the cycle appears during the traversal. One then obtains the number of k-cycles by adding the labels of all the k-cycles encountered. Table 3 displays the number of k-cycles in an m-machine cell. One can remark that this number rapidly grows so that it is impossible to find the best production cycle by enumeration of the cycle times of all k-cycles.

Table 3
Number of k-cycles in an m-machine cell

k	m					
	2	3	4	5	6	7
1	2	6	24	120	720	5 040
2	3	20	260	5 588	175 112	7 439 072
3	4	70	3 656	375 984	65 117 280	
4	6	300	60 648	29 222 424		
5	8	1 350	1 073 696			
6	14	6 580	19 847 316			
7	20	32 646				
8	36	166 620				
9	60	862 470				

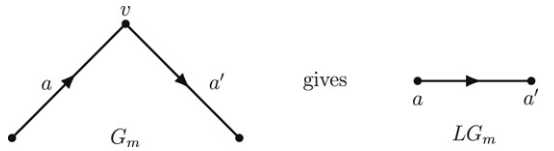

Fig. 5. Definition of an arc in LG_m.

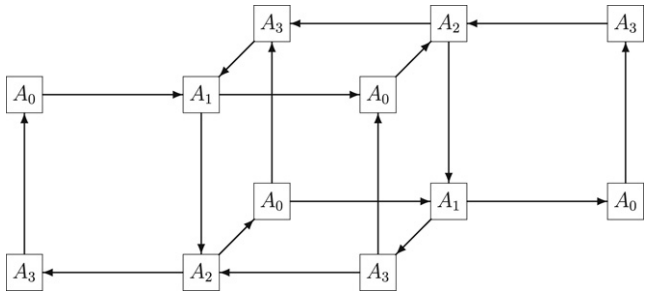

Fig. 6. The line-graph LG_3 of G_3.

To each graph G_m (with orientations) is associated a line-graph LG_m with orientations as follows:

- the vertices of LG_m correspond to the arcs of G_m;
- (a, a') is an arc of LG_m if and only if there exists, in G_m, a vertex v which is the tail of a and the head of a' as in Fig. 5.

Circuits of G_m and of LG_m are equivalent. Therefore, one can work with either graph. The graph LG_3 is represented on Fig. 6. Arcs of LG_m can be weighted by travel times of the robot or by some (unfortunately not all) waiting times. Therefore, a minimum mean circuit length in LG_m is a lower bound to the optimal cycle time.

Let G be an oriented graph with n vertices and a distance on the arcs. A *minimum mean circuit* C in G is a circuit which minimizes

$$\frac{\text{sum of the distances of the arcs of } C}{\text{number of vertices in } C}.$$

Finding a minimum mean circuit is a problem which can be solved in a time polynomial in the number of vertices of the graph (see e.g. [31,2]). Remark that the graph LG_m has an exponential number of vertices ($(m+3)2^{m-2}$). However, this approach has been used to prove many dominance results described in the following sections.

2.4. Special solved cases

In this section, we consider three very simple solved cases: a very slow or very fast robot (compared to processing times) and the 2-machine case.

Intuitively, when travel times are much larger than processing times, it can be interesting to carry out as few robot moves as possible. In this case, it is optimal to enter a part in the cell and to let the robot transfer this part on all the machines successively, waiting each time for the processing to complete. This is done by the 1-cycle $\pi_0 = (A_0 A_1 \dots A_m)$ also known as the *identity cycle*. Its cycle time is

$$T(\pi_0) = \sum_{h=0}^{m} \delta_{h,h+1} + \delta_{m+1,0} + 2(m+1)\epsilon + \sum_{h=1}^{m} p_h.$$

Table 4
Cycle function, $\mathcal{K}(m)$, in the unbounded case (problem (P1) in Section 2.1)

Travel	Production	2	3	4	$5 \le m \le 15$	$m \ge 16$
General		1	≥ 2			
Circular		1	?[a]		?	
Additive or constant		1		≥ 4[a]		
Additive regular	Balanced			1		?
Constant	Balanced				1	

[a] Means "even for the regular case".

Table 5
1-cycle complexity and performance in the unbounded case (problems (P2) and (P3) for $k = 1$ in Section 2.1)

Travel	Production	Complexity	Performance
General		NP-hard	4
General	Balanced	NP-hard	4
Circular		?	?
Additive/constant		Polynomial	1,5
Additive	Balanced	$O(1)$	?
Constant	Balanced	$O(1)$	1

On the other side, when the robot is very fast (compared to processing times), it is interesting to let it do many moves while other parts are being processed on the machines. In this case, the 1-cycle $\pi_d = (A_0 A_m A_{m-1} \dots A_1)$ also known as the *downhill cycle* is optimal. Its cycle time is

$$T(\pi_d) = \max\left(\sum_{h=0}^{m} \delta_{h,h+1} + \sum_{h=0}^{m-1} \delta_{h+2,h} + 2(m+1)\epsilon + \delta_{1,m}; \max_i (p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon)\right).$$

Note that if $T(\pi_d)$ is equal to the second term of the max operator, then the lower bound in inequality (7) is attained. Therefore, in this case (large processing times), π_d is optimal.

Another extreme problem is the 2-machine case. When the robot transfers a part from M_1 to M_2, both machines are empty. Moreover, between two consecutive occurrences of A_1, one has A_0A_2 or A_2A_0. Therefore, a k-cycle can be decomposed into k sequences of the two 1-cycles $A_1A_0A_2$ and $A_1A_2A_0$ and its cycle time is the sum of the cycle times of the sub-cycles composing it. Therefore, in this case, one-cycles are optimal (whatever the configuration of the cell is) and two 1-cycles are possible: the identity and the downhill permutation. Just choose the best one.

3. The unbounded case

In the seminal paper [40], the authors consider additive regular robotic cells with unbounded waiting times at machines and present the 1-cycle Conjecture. This section relates known results in the unbounded case, first describing the 9-year lifetime of the 1-cycle Conjecture for additive cells (Section 3.1). All results are then extended to the constant case (Section 3.2). Interest in 1-cycles and their simplicity induced research on their performance factor (Section 3.3). We then consider the well understood (but still partially open) regular balanced case for which the input is composed of only 4 numbers (Section 3.4).

On our way, we describe the state of problems (P1), (P2) and (P3) (defined in Section 2.1) for different configurations of unbounded robotic cells: Table 4 summarizes known values or bounds for the cycle functions $\mathcal{K}(m)$ depending on the cell configurations. Because of their simplicity, 1-cycles have generated a large amount of results. Table 5 shows the complexity of finding the best 1-cycles and the performance factor of 1-cycles.

3.1. The 1-cycle Conjecture for additive cells

In 1992, Sethi, Sriskandarajah, Sorger, Blazewicz and Kubiak [40] claim that the best production cycle can be achieved by a one cycle (*i.e.* $\mathcal{K}(m) = 1$ for any m) in a regular additive cell (called the 1-cycle Conjecture). In the same paper, the authors state that this conjecture is true for 2-machine cells. In 1997, Hall, Kamoun and Sriskandarajah [29] prove, rather technically, that 1-cycles dominate 2-cycles in regular 3-machine cells. In 1999, Crama and van de Klundert [21] extend this result to k-cycles proving that the 1-cycle Conjecture is valid for 3-machine additive cells (a shorter proof based on the state graph $G(3)$ and on inequalities (1) to (6) is given in [9]). Moreover, in regular additive 4-machine cells, 1-cycles again dominate 2-cycles [6].

In 1997, interest in 1-cycles is enforced when Crama and van de Klundert [19] prove that the best 1-cycle can be found in polynomial time. In [40], the authors prove that a 1-cycle is completely defined by a permutation of activities. Hence, the number of 1-cycles is $m!$. We consider, without loss of generality, that a 1-cycle π starts with activity A_0. 1-cycles are then of the form $\pi = (A_0, A_{i_1}, A_{i_2} \dots A_{i_m})$ where $(i_1, i_2 \dots i_m)$ is a permutation of $\{1, 2 \dots m\}$. Let us consider 1-cycles π belonging to the set of *pyramidal permutations*. We call $\pi = (A_0, A_{i_1}, A_{i_2} \dots A_{i_m})$ pyramidal if there is a p such that $1 \le i_1 < \dots < i_p = m$ and

$m > i_{p+1} > \cdots > i_m \geq 1$. In [19], the authors prove that pyramidal permutations dominate 1-cycles. They give an algorithm of complexity $O(m^3)$ for the determination of the best pyramidal permutation which is therefore also the complexity of finding the best 1-cycle.

Unfortunately, the 1-cycle Conjecture happened to be false for additive regular 4-machine cells. Indeed, in 2001, Brauner and Finke [11] described a 3-cycle that is strictly better than all 1-cycles for a regular additive 4-machine cell thus proving that $\mathcal{K}(4) \geq 3$ and reviving the question of the complexity of finding the best production cycle in robotic cells.

3.2. A metric unifying both the additive and the constant cases

All results presented in Section 3.1 are also valid for the constant case. Indeed, in [23], the authors claim that the graphical proof of the validity of the 1-cycle Conjecture for 3-machine additive cells in [9], also works for the constant case. We conjecture that the proof works for general 3-machine cells with travel times verifying $\delta_{12} + \delta_{23} + \delta_{40} \geq \delta_{24} + \delta_{13} + \delta_{02}$.

For the complexity of finding the best 1-cycle, define the *basic cycles* constructed as follow [22]:

- partition the activities $A_1, A_2 \ldots A_m$ into two sets V_1 and V_2;
- construct a sequence S of activities composed of A_0 followed by the activities of V_2 in decreasing order;
- insert sequentially the activities of V_1 in S in increasing order putting $A_i \in V_1$ just after A_{i-1} in S.

Let us illustrate this on an example: $m = 8$ and $V_1 = \{A_1, A_2, A_4, A_8\}$ and $V_2 = \{A_3, A_5, A_6, A_7\}$ make the 1-cycle $A_0A_1A_2A_7A_8A_6A_5A_3A_4$. All 1-cycles that can be constructed as described above belong to the set of basic cycles (of cardinality $2^m - m$). In [22], the authors prove that in a constant robotic cell, basic cycles dominate 1-cycles and that the best basic cycle can be found in polynomial time. Therefore, one has:

Theorem 1 (*[19,22]*). *In additive or constant robotic cells, the best* 1*-cycle can be found in polynomial time.*

In the quest for $\mathcal{K}(m)$, the 1-cycle Conjecture was replaced by Agnetis' Conjecture that claims that $\mathcal{K}(m) \leq m - 1$. This conjecture was again proved to be false for 4-machine cells:

Proposition 2 (*[12]*). *In a* 4*-machine cell, the* 4*-cycle*

$$C_4 = (A_0A_1A_0A_3A_4A_2A_1A_0A_3A_2A_1A_4A_3A_2A_0A_1A_4A_3A_4A_2)$$

strictly dominates all k-cycles for $k = 1, 2, 3$ *for the following instance:*

$m = 4; \quad \delta = 1; \quad \epsilon = 0; \quad p_1 = 0; \quad p_4 = 0;$
in the additive case : $p_2 = 10; p_3 = 10;$
in the constant case : $p_2 = 6; p_3 = 6.$

We conjecture that Proposition 2 is still valid with

- the following metric that generalizes the additive and the constant case: let δ_i be the time to travel between two machines with distances in the production process equal to i: $\delta_{kl} = \delta_{|l-k|}$, for $l, k = 0, 1 \ldots m + 1$ with the following assumptions

$$\delta_i \leq \delta_j \quad \text{for} \quad i \leq j \quad \text{and} \quad 0,4 < \frac{\delta_2}{\delta_1 + \delta_3} < 1$$

- the same instance generalized to $p_2 = p_3 = 3\delta_1 + 2\delta_2 + \delta_3$.

Preceding extensions of the properties of the additive case to the constant case raise two natural questions:

- Is there a metric which generalizes both cases keeping known results on the optimality and the complexity of 1-cycles?
- Is there a natural way to extend all proofs for the constant case (which seems simpler to study) to the additive case?

3.3. Performance of 1-cycles

Whenever the 1-cycle Conjecture is false, it is interesting to study the quality of 1-cycles since they are simple, easy to implement and the problem reduced to 1-cycles is polynomially solvable for most cases. Let us define the performance factor of 1-cycles, $\mathcal{P}(1)$, as in Section 2.1:

$$\mathcal{P}(1) = \max_{\text{instances}} \frac{\text{cycle length of the best 1-cycle}}{\text{cycle length of the best cycle in } S_{\mathcal{K}}}.$$

The 1-cycle π_d allows to find the following bounds:

$$\mathcal{P}(1) \le \left(2 - \frac{\delta_{0,1} + \delta_{m,m+1}}{\delta_{0,1} + \delta_{m,m+1} + \sum_{i=1}^{m-1} \delta_{i,i+1}}\right) \le 2 \quad \text{in the additive case [10,19];}$$

$$\mathcal{P}(1) \le \left(2 - \frac{2}{m+2}\right) \le 2 \quad \text{in the constant case [23];}$$

$$\mathcal{P}(1) \le 4 \quad \text{in the general case [25].}$$

For the additive regular case or for the constant case one has, $\mathcal{P}(1) \le 1.5$ [25]. For a regular additive 4-machine robotic cell, one can give this value with more precision: We know that $\mathcal{P}(1) \ge 16/15$ with the example presented in Proposition 2. With a rather technical study, one may show that $\mathcal{P}(1) \le 9/8$ and that this number is not tight.Finally, the reduced interval for $\mathcal{P}(1)$ in a regular additive 4-machine cell is given by [1.06666; 1.125].

3.4. Regular balanced cells

In this section, we consider balanced cells where an instance is given by four numbers:

- the number of machines, m,
- the travel time, δ (between consecutive machines in the additive case and between any two machines in the constant case),
- the loading/unloading time, ϵ,
- the processing time, p.

This case is interesting since it raises complexity problems: the complete description of a production cycle (as a k-cycle for instance) is not polynomial in the instance length (see [8] for a discussion on this topic). We shall discuss the complexity of finding the best 1-cycle and the optimality of 1-cycles for the constant (Section 3.4.1) and the additive (Section 3.4.2) cases.

3.4.1. Constant

For the constant balanced case, the problem is easy. We prove that, if $p < \delta$ then the identity cycle $\pi_0 = (A_0 A_1 \dots A_m)$ is optimal, otherwise, the downhill permutation $\pi_d = (A_0 A_m A_{m-1} \dots A_1)$ is optimal. If $p = \delta$, then both cycles perform equally well.

Theorem 3. *In the constant balanced case, one has $\mathcal{K}(m) = 1$ and the best 1-cycle can be found in constant time.*

Proof. Consider a k-cycle C_k. Its cycle time is composed of travel times, loading/unloading times and waiting times. Eq. (10) indicates that the total travel time of C_k is $2k(m+1)\delta - \sum u_i \delta$. Moreover, in C_k, each of the $k \times (m+1)$ activities induces a loading and unloading time of duration ϵ each. This leads to a total loading/unloading time of $2k(m+1)\epsilon$. Moreover, while executing a sequence $A_{i-1}A_i$, the robot transfers a part to M_i, waits at the machine during the process of the part and then transfers this part to machine M_{i+1}. Therefore, each $u_i = |A_{i-1}A_i|$ generates a waiting time of p. Hence the cycle time of the k-cycle C_k satisfies

$$T(C_k) \ge 2k(m+1)\delta + \sum_{i=1}^{m} u_i(p - \delta) + 2k(m+1)\epsilon.$$

Moreover, the cycle times of the identity cycle and of the downhill permutation are

$$T(\pi_0) = (m+2)\delta + mp + 2(m+1)\epsilon$$

$$T(\pi_d) = 2(m+1)\delta + 2(m+1)\epsilon + \max(0, p - (2m-1)\delta - 2(m-1)\epsilon).$$

If $p \ge (2m-1)\delta + 2(m-1)\epsilon$, then π_d achieves the lower bound given in inequality (9) and hence, π_d is optimal. For the following, we shall assume that $p \le (2m-1)\delta + 2(m-1)\epsilon$ so that $T(\pi_d) = 2(m+1)\delta + 2(m+1)\epsilon$.

If $p \ge \delta$ then $T(C_k) \ge 2k(m+1)\delta + 2k(m+1)\epsilon = kT(\pi_d)$ and π_d is at least as good as C_k.

If $p \le \delta$ then, since $u_i \le k$ and $p - \delta \le 0$, one has

$$T(C_k) \ge 2k(m+1)\delta + mk(p-\delta) + 2k(m+1)\epsilon = k(m+2)\delta + 2k(m+1)\epsilon + mkp = kT(\pi_0)$$

which proves that $\mathcal{K}(m) = 1$. A polynomial time algorithm for the best 1-cycle would return π_0 if $p < \delta$ and π_d if $p > \delta$ and π_0 or π_d for $p = \delta$.

3.4.2. Additive regular case

In additive regular balanced cells, the best 1-cycle can also be found in polynomial time [14]. We only consider the case $m \ge 4$, since for $m \le 3$, for the more general additive case, the 1-cycles are dominant and finding the best 1-cycle can be done in polynomial time.

Table 6
Dominance and complexity results for the no-wait additive case

m	Config.	$\mathcal{K}(m)$	Complexity
2	Add. reg.	1	$\mathcal{P}$ [1]
3	Add. reg.	2	$\mathcal{P}$ [1]
4	Add. reg.	≥ 3	Open
4	Reg. bal.	3	$\mathcal{P}$ [38]
≥ 5	Reg. bal.	$\geq m-1$	Open (1-cycles in pol. time but not optimal)

Consider the $\frac{m}{2}+1$ following 1-cycles for $m \geq 4$ and m even:

$$\pi_0 = (A_0A_1A_2 \ldots A_m)$$
$$\pi_\alpha = (A_0A_{\alpha+1}A_{\alpha+3} \ldots A_{m-\alpha-1}A_mA_{m-1}A_{m-2} \ldots A_{m-\alpha}A_{m-\alpha-2}A_{m-\alpha-4} \ldots A_{\alpha+2}A_\alpha A_{\alpha-1}A_{\alpha-2} \ldots A_2A_1) \quad \text{for } \alpha \in \left[1 \ldots \frac{m}{2}-1\right]$$
$$\pi_{m/2} = (A_0A_mA_{m-1} \ldots A_2A_1) = \pi_d.$$

Theorem 4 ([14]). *In an additive regular balanced cell, if m is even, the best 1-cycle is*

$$\begin{cases} \pi_0 & \text{if } 0 \leq p \leq \delta; \\ \pi_\alpha & \text{if } (4\alpha-3)\delta + 2(\alpha-1)\epsilon \leq p \leq (4\alpha+1)\delta + 2\alpha\epsilon;\ \alpha \in \left[1 \ldots \frac{m}{2}-1\right]; \\ \pi_{\frac{m}{2}} & \text{if } (2m-3)\delta + (m-2)\epsilon \leq p. \end{cases}$$

The case with m odd is similar. Consider the following 1-cycles for $m \geq 5$ and m odd :

$$\pi_0 = (A_0A_1A_2A_3 \ldots A_m)$$
$$\pi_0^1 = (A_0A_1A_3 \ldots A_{m-2}A_mA_{m-1}A_{m-3} \ldots A_4A_2)$$
$$\pi_0^2 = (A_0A_2A_4 \ldots A_{m-1}A_mA_{m-2}A_{m-4} \ldots A_3A_1)$$
$$\pi_\alpha^1 = (A_0A_{\alpha+1}A_{\alpha+3} \ldots A_{m-\alpha-2}A_mA_{m-1}A_{m-2} \ldots A_{m-\alpha-1}A_{m-\alpha-3}A_{m-\alpha-5} \ldots A_{\alpha+2}A_\alpha A_{\alpha-1}A_{\alpha-2} \ldots A_2A_1) \quad \text{for } \alpha \in \left[1 \ldots \frac{m-3}{2}\right]$$
$$\pi_\alpha^2 = (A_0A_{\alpha+2}A_{\alpha+4} \ldots A_{m-\alpha-1}A_mA_{m-1}A_{m-2} \ldots A_{m-\alpha}A_{m-\alpha-2}A_{m-\alpha-4} \ldots A_{\alpha+3}A_{\alpha+1}A_{\alpha-1}A_{\alpha-2} \ldots A_2A_1) \quad \text{for } \alpha \in \left[1 \ldots \frac{m-3}{2}\right]$$
$$\pi_{\frac{m-1}{2}} = (A_0A_mA_{m-1} \ldots A_2A_1) = \pi_d.$$

The permutations π_α^1 and π_α^2 have the same cycle time.

Theorem 5 ([14]). *In an additive regular balanced cell, if m is odd, the best 1-cycle is*

$$\begin{cases} \pi_0 & \text{if } 0 \leq p \leq \delta; \\ \pi_0^1 & \text{if } \delta \leq p \leq 2\delta; \\ \pi_\alpha^1 & \text{if } (4\alpha-2)\delta + 2(\alpha-1)\epsilon \leq p \leq (4\alpha+2)\delta + 2\alpha\epsilon;\ \alpha \in \left[1 \ldots \frac{m-3}{2}\right]; \\ \pi_{\frac{m-1}{2}} & \text{if } (2m-4)\delta + (m-3)\epsilon \leq p. \end{cases}$$

Therefore, for a given instance, one just has to determine α from m, p, δ, and ϵ to obtain the best 1-cycle. However, for additive regular balanced cells, the exact value of $\mathcal{K}(m)$ is not completely known: for $m \leq 3$, for the more general additive cells one has $\mathcal{K}(m) = 1$. For regular additive balanced cells, the detailed proof that 1-cycles are dominant can be found in [6] for $m = 4$ and [7] for $m \leq 6$. This last proof can be extended to $m \leq 15$ with a case by case study. However, the value of $\mathcal{K}(m)$ with $m \geq 16$ is still unknown.

4. The no-wait case

In this section, we review the no-wait case. Agnetis' conjecture, which claims that $\mathcal{K}(m) = m-1$, was first formulated for this case. Table 6 summarizes known results on the value of $\mathcal{K}(m)$ and on the complexity of finding the best production cycle for different values of m.

The no-wait case has historically been studied for additive regular cells. For $m \leq 3$, Agnetis' Conjecture was proved in [1]. For $m = 4$, it has been proved for the balanced case (equal processing times on all machines) in [38]. The idea of the proof is

Table 7
Optimal cycles for regular balanced 4-machine cells

p	Optimal cycle	Degree	Cycle length
$[0, 4\delta[$	$A_0A_1A_2A_3A_4$	1	$4p + 10\delta$
$[4\delta, 6\delta[$	$A_0A_1A_0A_2A_1A_3A_2A_4A_3A_4$	2	$5p/2 + 7\delta$
$[6\delta, 8\delta[$	$A_0A_1A_2A_3A_4$	1	$2p + 10\delta$
$[8\delta, 10\delta[$	$A_0A_3A_2A_1A_4A_3A_2A_4A_3A_0A_4A_1$	2	$5p/4 + 4\delta$
$[10\delta, 12\delta[$	$A_0A_4A_3A_1A_0A_4A_2A_1A_0A_3A_2A_1A_4A_3A_2$	3	$4p/3 + 14\delta/3$
$\geq 12\delta$	$A_0A_4A_3A_2A_1$	1	$p + 4\delta$

Table 8
Optimal cycles for regular balanced 5-machine cells

p	Minimal degree of an optimal cycle	Optimal cycle length
$[0, 4\delta[$	1	$5p + 12\delta$
$[4\delta, 8\delta[$	2	$6p/2 + 8\delta$
$[8\delta, 10\delta[$	3	$7p/3 + 20\delta/3$
$[10\delta, 12\delta[$	3	$6p/3 + 18\delta/3$
$[12\delta, 14\delta[$	4	$6p/4 + 20\delta/4$
$[14\delta, 16\delta[$	4	$5p/4 + 18\delta/4$
$\geq 16\delta$	1	$p + 4\delta$

to use the state graph G_4 and its line graph. The no-wait constraints allow, for each instance, to erase some forbidden arcs in the graph. In the reduced graph, structural properties allow to prove the desired result. Table 7 describes the optimal cycle for a regular balanced 4-machine cell. For simplicity, it considers the case $\epsilon = 0$. The first column gives the corresponding interval for the processing time p. Moreover, for $m \geq 4$, the cycle function verifies $\mathcal{K}(m) \geq m - 1$. This was proved in [36] using the $(m-1)$-cycle constructed below:

$$
\begin{array}{cccccccc}
A_0 & A_m & A_{m-1} & \dots & A_4 & A_3 & & A_1 \\
A_0 & A_m & A_{m-1} & \dots & A_4 & & A_2 & A_1 \\
A_0 & A_m & A_{m-1} & \dots & & A_3 & A_2 & A_1 \\
 & & & \vdots & & & & \\
A_0 & A_m & A_{m-1} & \vdots & A_4 & A_3 & A_2 & A_1 \\
A_0 & A_m & & \dots & A_4 & A_3 & A_2 & A_1 \\
A_0 & & A_{m-1} & \dots & A_4 & A_3 & A_2 & A_1 \\
 & A_m & A_{m-1} & \dots & A_4 & A_3 & A_2 & .
\end{array}
$$

For instance, for $m = 5$, this leads to the following 4-cycle:

$$A_0A_5A_4A_3A_1A_0A_5A_4A_2A_1A_0A_5A_3A_2A_1A_0A_4A_3A_2A_1A_5A_4A_3A_2.$$

Indeed, while executing this cycle, the robot transports $m - 1$ parts through the cell, but does not have the time (because of the no-wait constraint) to transport an additional part (which would have given the 1-cycle π_d). Using this idea, in [39], a conjecture is given on a best (optimal and with minimum degree) k-cycle in a balanced cell. This conjecture, which goes a step further for solving Agnetis' Conjecture, is still open but some special cases have been solved. Table 8 summarizes the open cases (grey cells) in a regular balanced 5-machine cell (again with $\epsilon = 0$).

In [39], it is also proved that, for each value of $1 \leq k \leq \lfloor \frac{m+1}{4} \rfloor$, there exists an optimal k-cycle that strictly dominates all cycles with smaller degree. This implies that $\mathcal{K}(m)$ can not be bounded by a constant.

In the additive no-wait case, finding the best 1-cycle or 2-cycle can be done in polynomial time [34,17]. However, the preceding discussion indicates that 1- or 2-cycles are not optimal for $m \geq 4$. Therefore, the complexity of finding the optimal production cycle is still an open problem. If Agnetis' Conjecture is true, this problem is reduced to finding the best k-cycle with $k \leq m - 1$.

Some extensions of this problem have been studied: In [37], the combined case with some machines with no-wait constraints and others with unbounded waiting times is considered. For balanced 2- and 3-machine cells the complete dominance table is known. For this case, $\mathcal{K}(2) = 1$ and $\mathcal{K}(3) = 2$ as for the no-wait case.

In [16,33], the authors present algorithms for finding the best k-cycle in the no-wait case for identical parts. Those algorithms are polynomial in m but exponential in k.

5. Open questions

This document has presented an overview of the scheduling problem of identical parts in a robotic cell. In this section, we list the main questions that are still open. For unbounded waiting times, the 3-machine case is completely settled. Therefore, one can concentrate on the remaining questions on the 4-machine case (hoping that it will be possible to generalize the proofs to m machines):

- Prove that $\mathcal{K}(4) = 4$ in the additive regular or in the constant case: It is only known that, for certain instances, some 4-cycles strictly dominate cycles with smaller degree, but the dominance of 4-cycles is not proved.
- Determine the exact performance factor of 1-cycles in a 4-machine cell.

More general questions also remain for additive or constant cells:

- Find the link between additive and constant cells (same results) or a metric generalizing them with the same properties (complexity of finding the best 1-cycle, identical $\mathcal{K}(m)$ for all sub-configurations…).
- Determine lower and finite upper bounds for $\mathcal{K}$.
- Determine the complexity of finding the best k-cycle with $k \leq \mathcal{K}(m)$ in an m-machine cell.

For special configurations, the two main questions are:

- Determine $\mathcal{K}(3)$ for circular cells: little work has been done on this configuration and even the 3-machine case is still open. This is linked to the preceding question that tries to generalize the metrics.
- Prove that $\mathcal{K}(m) = 1$ for additive regular balanced cells and $m \geq 16$; even for this very simple (4 number) problem, the dominance of 1-cycles is not proved.

The no-wait case seems to be easier to settle since the problem is much more constrained than the case with unbounded waiting times. However, the problem of finding an optimal cycle is still open. The simpler configuration is the balanced problem (with equal processing times on all machines). Therefore, we suggest to start with this case for the following question and then to extend it to the general case.

- Prove (or refute) Agnetis' conjecture. This reduces to proving that $\mathcal{K}(m) \leq m - 1$.

References

[1] A. Agnetis, Scheduling no-wait robotic cells with two and three machines, European Journal of Operational Research 132 (2) (2000) 303–314.
[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows, Theory, Algorithms, and Applications, Prentice Hall, New Jersey, 1993.
[3] M.S. Akturk, H. Gultekin, O.E. Karasan, Robotic cell scheduling with operational flexibility, Discrete Applied Mathematics 145 (3) (2005) 334–348.
[4] C.R. Asfahl, Robots and Manufacturing Automation, John Wiley & Sons, New York, NY, 1985.
[5] C. Bloch, Contribution à l'ordonnancement dynamique de lignes de traitement de surface, Thèse de doctorat, Université de Franche-Comté, U.F.R. Sciences et Techniques, Besançon, 1999.
[6] N. Brauner, Ordonnancement dans des cellules robotisées, Thèse de doctorat, Université Joseph Fourier, Grenoble, France, 1999.
[7] N. Brauner, Y. Crama, G. Finke, One-unit production cycles in balanced robotic cells, in: Proceedings IEPM'01, International Conference on Industrial Engineering and Production Management, Quebec City, Canada, 2001, pp. 508–518.
[8] N. Brauner, Y. Crama, A. Grigoriev, J. van de Klundert, A framework for the complexity of high-multiplicity scheduling problems, Journal of Combinatorial Optimization 9 (2005) 313–323.
[9] N. Brauner, G. Finke, On a conjecture about robotic cells: New simplified proof for the three-machine case, Journal of Information Systems and Operational Research - INFOR: Scheduling in Computer and Manufacturing Systems 37 (1) (1999) 20–36.
[10] N. Brauner, G. Finke, Optimal moves of the material handling system in a robotic flow-shop, in: Proceedings IEPM'99, International Conference on Industrial Engineering and Production Management, vol. 1, Glasgow, United Kingdom, 1999, pp. 409–417.
[11] N. Brauner, G. Finke, Cycles and permutations in robotic cells, Mathematical and Computer Modelling 34 (5–6) (2001) 565–591.
[12] N. Brauner, G. Finke, Robotic cells: Configurations, conjectures and cycle functions, in: Operations Research 2005, Bremen, Germany, 2005.
[13] N. Brauner, G. Finke, C. Gueguen, Flow-shop robotisé monoproduit avec stockage, European Journal of Automation / Journal Européen des Systèmes Automatisés - APII-JESA 32 (7–8) (1998) 875–891.
[14] N. Brauner, G. Finke, W. Kubiak, Complexity of one-cycle robotic flow-shops, Journal of Scheduling 6 (4) (2003) 355–371.
[15] N. Brauner, B. Vettier, Testing module for cyclic robotic cells, in: Proceedings MCPL'2000, Second Conference IFAC/IFIP/IEEE Management and Control of Production and Logistics, vol. 2, Grenoble, France, 2000, pp. 723–728.
[16] A. Che, C. Chu, F. Chu, Multicyclic hoist scheduling with constant processing times, IEEE Transactions on Robotic and Automation 18 (1) (2002) 69–80.
[17] A. Che, C. Chu, E. Levner, A polynomial algorithm for 2-degree cyclic robot scheduling, European Journal of Operational Research 145 (1) (2003) 31–44.
[18] Y. Crama, V. Kats, J. van de Klundert, E. Levner, Cyclic scheduling in robotic flowshops, Annals of Operations Research: Mathematics of Industrial Systems 96 (2000) 97–124.
[19] Y. Crama, J. van de Klundert, Cyclic scheduling of identical parts in a robotic cell, Operations Research 45 (6) (1997) 952–965.
[20] Y. Crama, J. van de Klundert, Robotic flowshop scheduling is strongly NP-complete, in: O.J. Vrieze, W.K. Klein Haneveld, L.C.M. Kallenberg (Eds.), Ten Years LNMB, CWI Tract 122, Amsterdam, The Netherlands, 1997, pp. 277–286.
[21] Y. Crama, J. van de Klundert, Cyclic scheduling in 3-machine robotic flow shops, Journal of Scheduling 2 (1999) 35–54.
[22] M. Dawande, C. Sriskandarajah, S. Sethi, On throughput maximization in constant travel-time robotic cells, Manufacturing and Service Operations Management 4 (4) (2002) 296–312.
[23] M. Dawande, H.N. Geismar, S.P. Sethi, C. Sriskandarajah, Sequencing and scheduling in robotic cells: Recent developments, Journal of Scheduling 8 (5) (2005) 387–426.
[24] G. Finke, C. Gueguen, N. Brauner, Robotic cells with buffer space, in: Proceedings Conference of the European Chapter on Combinatorial Optimization, ECCO IX, Dublin, Ireland, 1996, page 9 pages unnumbered.
[25] H.N. Geismar, M. Dawande, C. Sriskandarajah, Approximation algorithms for k-unit cyclic solutions in robotic cells, European Journal of Operational Research 162 (2) (2005) 291–309.
[26] H.N. Geismar, S.P Sethi, J.B. Sidney, C. Sriskandarajah, A note on productivity gains in flexible robotic cells, International Journal of Flexible Manufacturing Systems 17 (1) (2005) 5–21.
[27] H. Gultekin, M.S. Akturk, O.E. Karasan, Cyclic scheduling of a robotic cell with tooling constraints, European Journal Operational Research 174 (2) (2006) 777–796.
[28] H.G. Hall, H. Kamoun, C. Sriskandarajah, Scheduling in robotic cells: Complexity and steady state analysis, European Journal of Operational Research 109 (1998) 43–65.
[29] N.G. Hall, H. Kamoun, C. Sriskandarajah, Scheduling in robotic cells: Classification, two and three machine cells, Operations Research 45 (3) (1997) 421–439.
[30] I. Ioachim, E. Sanlaville, M. Lefebvre, The basic cyclic scheduling model for robotic flow shops, INFOR 39 (3) (2001) 257–271.
[31] R.M. Karp, A characterization of the minimum cycle mean in a diagraph, Discrete Mathematics 23 (1978) 309–311.
[32] V. Kats, E. Levner, Cycle scheduling in a robotic production line, Journal of Scheduling 5 (1) (2002) 23–41.

[33] V. Kats, E. Levner, L. Meyzin, Multiple-part cyclic hoist scheduling using a sieve method, IEEE Transactions on Robotic and Automation 15 (4) (1999) 704–713.
[34] E. Levner, V. Kats, V.E. Levit, An improved algorithm for cyclic flowshop scheduling in a robotic cell, European Journal of Operational Research 97 (1997) 500–508.
[35] R. Logendran, C. Sriskandarajah, Sequencing of robot activities and parts in two-machine robotic cells, International Journal of Production Research 34 (12) (1996) 3447–3463.
[36] F. Mangione, Ordonnancement des ateliers de traitement de surface pour une production cyclique et mono-produit, Ph.D. Thesis, Institut National Polytechnique de Grenoble, 2003.
[37] F. Mangione, N. Brauner, B. Penz, Balanced hoist scheduling problem with unbounded or zero-width processing windows, European Journal of Automation / Journal Européen des Systèmes Automatisés - APII-JESA 37 (3) (2003) 391–404.
[38] F. Mangione, N. Brauner, B. Penz, Flow shop robotisé à quatre machines sans attente, in: Conférence Francophone de MOdélisation et SIMulation, MOSIM 03, Toulouse, France, 2003. pp. 542–545.
[39] F. Mangione, N. Brauner, B. Penz, Optimal cycles for the robotic balanced no-wait flow shop, in: Proceedings IEPM'03, International Conference of Industrial Engineering and Production Management, vol. 2, Porto, Portugal, 2003, pp. 539–547.
[40] S.P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, W. Kubiak, Sequencing of parts and robot moves in a robotic cell, International Journal of Flexible Manufacturing Systems 4 (1992) 331–358.
[41] S.P. Sethi, J.B. Sidney, C. Sriskandarajah, Scheduling in dual gripper robotic cells for productivity gains, IEEE Transactions on Robotics and Automation 17 (3) (2001) 324–341.
[42] Q. Su, F.F. Chen, Optimally sequencing of double-gripper gantry robot moves in tightly-coupled serial production systems, IEEE Transactions on Robotics and Automation 12 (1) (1996).